映像コンテンツへの拡張プログラム埋め込みパラダイムの再検討

栗原一貴[†1†2]　　橋本美香[†1]

本論文では代表的な時系列コンテンツの一つである映像コンテンツに対し，時刻情報の付与されたスクリプト言語を埋め込むことにより，映像と連動したプログラムを駆動しエンタテインメント価値を高める手法の再検討を行う．具体的には YouTube 等の映像コンテンツに対し公式にサポートされている字幕形式である srt ファイルの形で Javascript を埋め込み，映像のタイミングに合わせて外部 Web サービスや IoT 機器と連携することを容易に行えるフレームワークの開発を行った．これは Adobe Flash などが旧来的に実現してきたプログラミングパラダイムの再活用である．我々は種々の具体例により本手法の有効性を示し，問題点と今後の展望について整理した．

Revisiting A Programming Paradigm of Embedding Codes in Video Contents

KAZUTAKA KURIHARA[†1†2]　MIKA HASHIMOTO[†1]

In this paper we revisit a programming paradigm of embedding time-stamped scripts in videos, which are typical time series contents, to increase the entertainment value by executing the scripts in conjunction with the videos. Specifically, we developed a framework of embedding Javascript codes in YouTube videos as a srt file, which is an official format for describing subtitle information of a video. Users can easily develop a video-oriented application involving external web services and IoT devices by executing the codes in conjunction with the video. This is a revitalization of a traditional programming paradigm that was once hired by the framework such as Adobe Flash and has been widely used. We demonstrate the effectiveness of the framework by showing various examples of use cases built on it and discuss the future challenges.

1. はじめに

IoT (Internet of Things) という言葉で表わされるように，身の回りの様々なモノがインターネットに接続しインテリジェントに振る舞い我々の生活を豊かにする社会が到来しつつある．一般市民が趣味の工作として IoT 機器を制作するための要素技術も充実してきており，Intel Edison，Raspberry pi などの小型・安価な汎用コンピュータや，Sony MESH や webmo のように専門知識をほとんど必要とせずに気軽に IoT システム開発を行えるプロダクトも登場してきている．

そのような今日において，エンタテインメント用途のプログラミングは二極化している．一つは全てをことこまかにプログラミングした情報システムである．商用のゲームなどはこれに属する．多様なことが実現できる反面，システム開発のコストは大きい．一方，もう一つは IF-THEN ルールに単純化させたプログラミングである．IFTTT や myThings, および Sony MESH などは「もし〜がおきたら，〜する」という形のプログラミングを GUI 等で行うことによって，スマートフォンの機能や有名 Web サービスを連携したアプリの気軽な開発を可能にしている．

本論文は，人類がこれまで慣れ親しんでいた時系列コンテンツに対し現代的な機能拡張方法を検討するものである．ここで時系列コンテンツとは，提示される情報の順番が基本的には決まっていて，かつ比較的長い時間継続するコンテンツを指すこととする．音と動画像からなる映像コンテンツやプレゼンテーションなどがこれに該当する．これらの扱う情報は一方通行的にユーザに届けられ，またモダリティが限定されていたが，近年の各種 Web サービスや IoT 機器の普及により，時系列コンテンツの進行に連動してエンタテインメント価値を高められる可能性が広がった．たとえば映像共有サービス YouTube が，全天周カメラの映像に対応し，視聴するデバイスを動かすことによって視野をインタラクティブに変更できるようになったことはそのような事例の一つであり，また我々がこれまでに発表した，プレゼンテーションの進行に合わせて予め準備した内容の Twitter の投稿を行う「ぴぴつい[1]」や，映像に応じて物理的な開放状態を調整する IoT 機器である「開放度調整ヘッドセット[2]」も時系列コンテンツを拡張するシステムの事例として挙げることができるだろう．

しかし可能性が拓かれた一方で，現状ではそのような複合的な時系列コンテンツを開発しようとした場合の汎用的な仕組みが不足している．それほどのインタラクション頻度は無いにもかかわらず，汎用のプログラミング環境を用いた完全にゼロからの開発を行うのは煩雑であるし，

†1 津田塾大学
Tsuda College
†2 Diverse 技術研究所
Diverse Institute of Technology

IFTTT などのような IF-THEN ルール型の簡易プログラミング環境は時系列コンテンツ拡張には対応していない．

そこで我々は，かつて Adobe Flash などで標準的に用いられていた，時系列コンテンツに連動した任意のプログラムをそのコンテンツの時系列上に埋め込み，鑑賞時にタイミングよくプログラムを実行するパラダイムを再検討し，現代的な形式での開発環境を再提案する．既存の時系列コンテンツ資源を活用し，低コストで多少の機能拡張をしたいと思った場合，ほとんどの制御は「時刻 X になったら処理 Y を実行する」という，互いに独立した WHEN-THEN 型ルールがコンテンツ中に比較的疎に散在する形になる．大部分の計算と通信がほぼリアルタイムもしくはノンブロックで行える現代の計算機環境においては，実際にプログラム実行にどれだけ時間がかかるかはあまり問題にならない場合が多いため，時刻情報とともに実行すべきプログラムを埋め込むのはごく自然な方法であり，開発にも適する方法である．またコンテンツ自体にプログラムが付随することによって一元的な開発と管理，配布が容易に行える点も特徴である．

本稿では，時系列コンテンツへの埋め込みスクリプトによる機能拡張について，YouTube 等の映像コンテンツを対象とし, YouTube の字幕機能として Javascript を埋め込んでエンタテインメント性を拡張する手法にしぼり報告する．プレゼンテーションへのとりくみについては次報で報告する．

本稿の構成は以下の通りである．まず次節で関連研究について述べる．その後提案システムの実装について述べ，多様な運用例を示すことで提案手法の有効性を示す．その後現状の問題点および今後の展望を議論しまとめとする．

2. 関連研究

時系列コンテンツをスクリプトにより拡張する研究は新規なものではない. Adobe Flash は時系列アニメーションコンテンツのオーサリングにおいて，指定された任意の時刻に実行されるプログラム（ActionScript）を埋め込む機能を実装しており，長年にわたり標準的に用いられてきた．その柔軟な拡張性の一方で，タイムラインと紐付ける形でプログラムを記述するパラダイムには以下の問題があったと分析される．

1． コンテンツがインタラクティブになればなるほどタイムライン通りに進行しないので，タイムラインから「実行の順と時刻を保証するもの」という意味合いが失われる．そのような環境でどのようなプログラム要素も便宜的にどこかの時刻に割り当てなければならないのは直感に反し，開発がやりにくい．

2． 全てのプログラムは，ある開始時刻と終了時刻を付与されて記述される．しかしそのプログラムがその時間内でちょうど実行終了する保証がないため，時間指定が便宜的になってしまう．

3． コンテンツのほとんどを唯一の ActionScript ブロックが占める場合，タイムラインがほぼ不要となる．

特に 1 は，コンテンツにインタラクティブ性をいかに付与するかが中心的課題だった当時において重要な課題だったと考えられる．Kurihara らによる AfterThought は，flash のもつ 1 および 2 の問題に対応し，プログラムを埋め込む拡張として，絶対時刻がなく，順序関係だけが意味を持つタイムラインを提案し，制約解消を用いた柔軟な UI を提案している[3]．また，3 についてはのちにタイムラインを用いずに純粋に ActionScript のみでコンテンツ開発を行う環境である Adobe Flex（のちに Apache Flex）が登場して解決された．

本研究では，現代に即したプログラミングのパラダイムとして再度，絶対時刻つきタイムライン連動型のプログラム埋め込み方法の可能性について検討を行う．以前と比較して社会状況が好転した点として，以下が挙げられる．

- タブレット・スマートフォンなどの普及などにより，人々がエンタテインメントコンテンツに接する際の態度がより受動的なものになり，コンテンツに要求されるインタラクションの強度が必ずしも高くないものが認知されるようになってきた．
- 渡邊の「融けるデザイン」[4]にあるように，現代はコンテンツがユーザの時間を奪う時代であり，ユーザは奪われる時間に敏感にならざるを得ない．インタラクティブ性の少ない時系列コンテンツは，ユーザから奪う予定の時間を事前に提示しやすいため，インフォームドコンセントの観点から優れている．
- 計算機と通信の技術革新により，おおよその計算と通信がほぼリアルタイムもしくはノンブロックで行えるようになり，実際にプログラム実行にどれだけ時間がかかるかはあまり問題にならない場合が増えた．
- IFTTT や myThings などの登場により, 既存のサービスに「少しだけ」独自要素を追加するプログラミングが受け入れられるようになってきた．時系列コンテンツへの「少しだけ」の独自要素の追加であれば，タイムライン上に疎にプログラムが埋め込まれる程度であるため, タイムラインというメタファーは破綻しない．
- 時系列コンテンツについては，YouTube やニコニコ動画などの動画共有サービスの普及により，独自要素を追加したいコンテンツは豊富に存在し，流通も容易になった．

現在，そのような取り組みとして既に実用化されている事例として，ホラー映画と観客のスマートフォンを連動させ，鑑賞中に実際に電話が着信通知する演出などを盛り込んだ「貞子 3D2 スマ 4D」[5]，主にポルノ映像と性家電を連動させる規格である+1D[6]などが挙げられる．我々はこのような映像連動コンテンツをエンドユーザ・プログラマ

が気軽に開発できるフレームワークを提供する．

後藤らの Songle API[7]や Kato らの TextAlive[8]は YouTube やニコニコ動画における音楽つき映像を拡張するための道具立てとして，音楽コンテンツの解析情報やブラウザ上でのスクリプト言語の開発・共有を可能にする機能を提供している．我々の提案した開放度調整ヘッドセットにおける音楽コンテンツとの連携動作は Songle API の活用により実現している．また Stepmania Gimmic[9]は音楽ゲームを Lua スクリプトにより拡張するものである。しかしこれらは音楽つき映像コンテンツを対象としており，それ以外の一般の映像コンテンツを扱うものではない．

中村ら[10]および松田ら[11]は YouTube の映像再生について，音量，再生速度の調整，効果音，視覚効果の追加等のカスタマイズをブラウザ上で行い，他者と共有するものである．本研究はこれらのように個人的な映像カスタマイズも支援できる点では共通性があるが，Javascript による任意のコード記述が可能であるため，プログラミングの知識は多少必要になるものの映像の再生コントロールの枠を超えたカスタマイズ，例えば IoT 機器との連動やインタラクティブ性の付与等が可能である．

ほかにも IoT 時代に即した新しいプログラミングパラダイムの提案として，以下の研究が挙げられる．加藤らによる f3.js は，IoT 関連システム開発で問題となる，ハードウェア設計とソフトウェア設計を統合的に開発するための環境である[12]．馬場らによる babascript[13]，および橋本らによる goldfish[14]は，人間を実行主体として扱うプログラミング，および実世界を対象としたプログラミングを行う基盤として提案されている．

なお，我々が提案する，既存時系列コンテンツへの「少しだけ」の独自要素の追加は，過去に著者らがゲーミフィケーションの周辺概念として提唱した Toolification of Games[15]と共通性を持つ. すなわちある目的を達成する際，専用の情報システムをゼロから開発するのではなく，一方で目的の達成を助ける外部エンタテインメント要素を小規模に追加する（いわゆるゲーミフィケーション）のでもなく，既存の優良エンタテインメントコンテンツの中に寄生させる形で追加するという点である．

3. 提案システム

我々は映像連動コンテンツをエンドユーザ・プログラマが気軽に開発できるフレームワークを提案する．システム構成図を図 1 に示す． Web ブラウザ上の YouTube 等の映像プレイヤーで再生される映像に同期して，各種 Web サービスやスマートフォン，および各種 IoT 機器と通信し映像コンテンツを拡張する．拡張のためのプログラムは YouTube の映像コンテンツに対し公式にサポートされている字幕形式である srt ファイルの形で Javascript を埋め込み，映像のタイミングに合わせて実行することで実現する．以下に特記すべき要素について詳細に説明する．

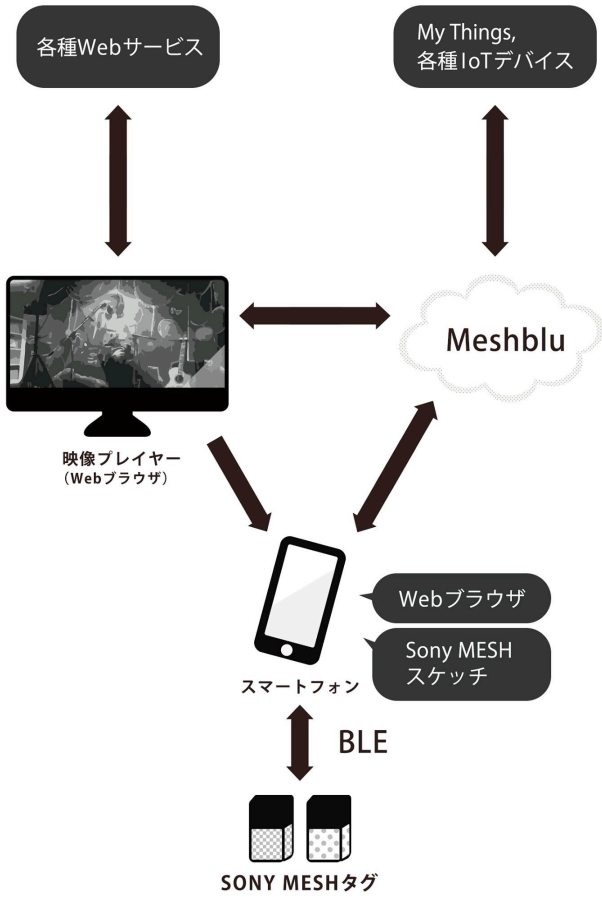

図 1 システム構成

Figure 1 System Configuration.

3.1 srt.js 形式のスクリプト

srt.js 形式は，映像に対し付与される字幕情報のフォーマットとして一般的に普及している srt 形式によって，時刻に連動した Javascript のプログラム実行を指示するテキストフォーマットである．srt は図 2 のように，通し番号（1，5，9 行目），字幕開始時刻・終了時刻（2，6，8 行目），字幕テキスト（3，7，11 行目）を一つのブロックとして記載したもので，空行が区切り記号として用いられている．字幕テキストは改行して複数行にわたっても問題ない．したがって空行さえ用いなければ，字幕テキスト記載部にほぼ任意の Javascript の記述が可能である．字幕と同様に指定のタイミングで指定の Javascript コード群を実行することを指示するこのフォーマットを srt.js 形式と名付ける．図 3 は開始 1 秒後に"Once upon a time (以下略)"とアラート表示する処理などを含む srt.js 形式のファイルの例である．

srt.js 形式のファイルは YouTube 映像の所有者が公式の字幕情報として登録できるため，管理と配布が容易である．その他の srt.js の仕様を以下にまとめる．

- 再生中の映像と連動するため, html5 の動画オブジェクトや YouTube の iframe 埋め込み動画プレイヤーオブ

ジェクトを player という名前で参照できる．

- 今どの字幕区間（すなわち，あるプログラム要素が実行される区間）にいるかを示すグローバル変数 index をもつ．どの区間にも入っていない場合は-1 が入る．
- あるイベントリスナが発火した際に，映像の再生箇所に対応した関数を実行するための仕組みである indexedFunction が用意されている．これにより，たとえばスマートフォンをシェイクするなどの同一のユーザ入力に対して，映像の再生時刻に応じた別の処理を行うことが可能である．
- 外部 Javascript を読み込むヘルパー関数である loadScript，および後述する IoT 機器連携機能が予め用意されている．

```
1
00:00:01,000 --> 00:00:04,000
Once upon a time, there was a boy.

2
00:00:05,000 --> 00:00:08,000
He lived in a small city.

3
00:00:10,000 --> 00:00:13,000
One day, he met a girl.

```

図 2　srt ファイルの例

Figure 2　An Example of srt format.

```
1
00:00:01,000 --> 00:00:04,000
// srt.js example
alert("Once upon a time, there was a boy.");

2
00:00:05,000 --> 00:00:08,000
console.log("He lived in a small city.");

3
00:00:10,000 --> 00:00:13,000
var msg = "One day,";
msg = msg + " he met a girl.";

```

図 3　srt.js ファイルの例

Figure 3　An Example of srt.js format.

3.2　映像と連動してプログラムを実行する 2 種類の映像プレイヤー

映像プレイヤーとして，(1) YouTube の iframe 埋め込み API を用いた静的なウェブサイトとして実装されたものと，(2) chrome 拡張を用いて YouTube および Amazon Prime Video の公式サイト訪問時に起動するものの 2 種類を用意した．(1)はスマートフォンを含む任意のブラウザで事前準備なく実行可能であり，プレイヤーを表示するウェブサイト全体のデザインができる点が利点であるが，iframe 埋め込み API を通じた映像の制御は再生速度の調整が任意の速度にできないなどの欠点がある．(2)は html5 の video 要素に連動した srt.js の実行を行うため，よりきめ細かい映像制御ができる点，および Amazon Prime Video にも対応している点が利点であるが，chrome 拡張を用いるため，インストールの手間がかかり，また実行環境として PC の chrome ブラウザが必要な点が欠点である．

(1)は https://srtjs.azurewebsites.net/?v=xxx にアクセスするだけで使用可能である．ただし xxx の部分には YouTube の映像の識別子である 11 桁の文字列が入る．本サイトはクライアントサイド Javascript のみで実装されている静的なサイトであるため，必要に応じてユーザが自身の管理するウェブサイトに組み込むことも容易である．(2)については chrome web store から配布予定である．

なお(1)(2)を用いて，既に他者によって公開されている映像に対してユーザが srt.js を連動させたい場合や，(2)で srt 形式に対応していない Amazon Prime Video を対象にする場合は，ユーザが作成した srt.js ファイルをネット上でアクセス可能な状態にしたうえで以下のように GET パラメータである surl を加えることで活用可能である．

https://srtjs.azurewebsites.net/?v=xxx&surl=(srt.js の URL)

3.3　Meshblu

映像が再生されている web ブラウザと IoT 機器等を連携するための通信基盤として，IoT 通信プラットフォームである Meshblu を採用した．これは http(s) REST, web socket, MQTT などの様々なプロトコルで IoT 機器間の通信を仲介するサーバである．開発元である Octblu は誰でもすぐに利用できる Meshblu サーバを公開しており，それを活用可能である．また，IDCF クラウドを使用すると，Meshblu サーバを通じて IFTTT と類似のサービスである myThings と相互通信可能になるため，多様な機器，多様なサービスとの連携が容易に行える．srt.js では，http(s) REST API もしくは socket.io で Meshblu と接続する．

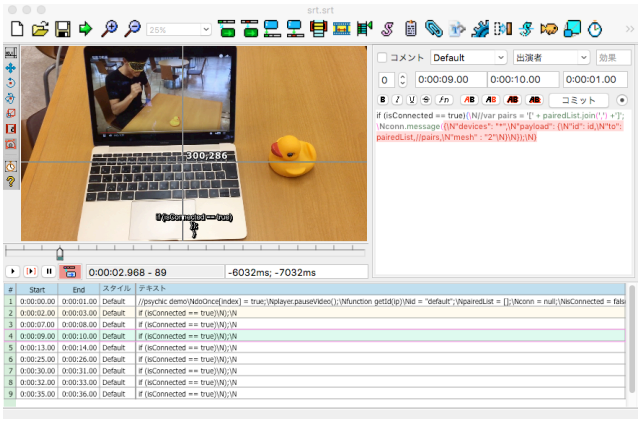

図 4　Aegisub

Figure 4　Aegisub.

3.4 映像に対する字幕情報として Javascript を記述するエディタ

基本的には srt.js を記述するコードエディタとしては任意のテキストエディタが活用可能である．映像を見ながら実行タイミングを調整しつつコーディングを行うためには，srt 作成を支援する字幕テキストエディタを活用するとよい．図 4 はそのような用途で活用可能な，オープンソースの字幕エディタ aegisub である．

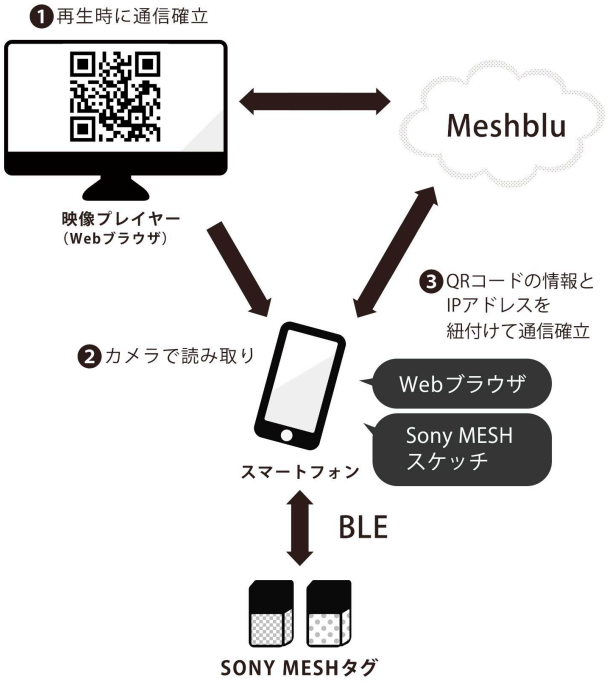

図 5　QR コードを用いた連携

Figure 5　Establishing a connection between devices using a QR code.

3.5 MESH 用の開発補助ライブラリ

気軽な IoT システム開発を実現可能な MESH には，Javascript による機能拡張をおこなえる SDK が準備されている．これを用いて，Meshblu と通信し subscribe（Meshblu への情報送信）および publish（Mesublu からの情報受信）が可能な拡張タグを実装した. Meshblu を経由した myThing と MESH との連携手法は既に myThings 上で公開されているが, ポーリング周期が遅いため最大 15 分程度の通信の遅れがある．開発したライブラリは送受信ともにこの遅れを解決し，次項でのべる動画との連携機能を備えたものである．

3.6 鑑賞中の動画と IoT 機器を連動させる機構

鑑賞中の動画と IoT 機器を安全に連携するため，QR コードおよびスマートフォンを用いた方式を実装した．これは映像の再生時にその再生を一意に指定する ID と Meshblu への接続情報を含んだ QR コードを表示するもので，スマートフォンからカメラで読み取った URL にアクセスすると, 動画との相互通信が確立する仕組みである（図 5）. srt.js 側では接続が確立した端末の承認リストを管理しており，接続確立後は承認された任意の端末間の通信が可能となる．ここでさらに，接続が確立したスマートフォン上で MESH のスケッチ（プログラム）が動いている場合は，スマートフォンの IP アドレス情報を照合することで鑑賞中の動画と MESH のスケッチとの通信も確立される．

4. ユースケースシナリオ

本節では提案システムの有効性を実証するために著者らが製作した映像連動コンテンツのプロトタイピング事例を列挙する．これらは一部を除きウェブサイトで公開されている（https://sites.google.com/site/qurihara/home/srtjs）.

4.1 ユーザの入力に基づく映像の再生制御

1. MESH のボタンタグによる動画の再生と一時停止.
2. 顔検出ライブラリを用いた,「見ている時だけ再生される映像」.
3. GPS および地図 API を用いた，映像による道案内.
4. ヘッドラインニュースにおいて，冒頭のニュース見出し部の各項目を閲覧時にスマートフォンをシェイクするとその項目の本編に移動する.
5. ビブリオバトルの映像において，紹介されている書籍に興味を持った際にスマートフォンをシェイクするとその書籍を購入できる Amazon サイトへと誘導される.
6. MESH の加速度タグが設置されたティーカップを頭に載せた状態で，それを落とさないようにラジオ体操を行うゲーム.

4.2 映像と同期した表現の拡張

1. “Happy birthday to you”の歌における“Happy birth day dear XX”の XX の部分において, 予め入力した名前を合成音声により読み上げる.
2. QR コードにより通信を確立させた全てのスマートフォンに対し，新年のカウントダウン映像に同期して，“Happy new year!”と表示させる.
3. 新年のカウントダウン映像に同期して，MESH の GPIO タグに接続された点火装置により花火を打ち上げる.
4. 登場人物が念じる映像を再生すると，付近にあるアヒルの置き物が動く「超能力映像」を MESH の GPIO タグに振動モータを接続して実現する.
5. 「スパイ大作戦」の「なお，この録音は自動的に消滅する」という台詞に同期して，ブラウザが強制終了する.
6. 「いないいないばあ」を動物キャラクタが行う子供向け映像を webmo に載せたスマートフォン上で再生し，実際にキャラクタが振り向くようにする.

7. 子供向けお笑い映像において，笑いがおこるタイミングで写真撮影する．

5. 活用事例とフィードバック

本節では提案フレームワークが第三者に活用され，フィードバックを得られた事例を 2 例報告し，提案システムの定性的評価を行う．

5.1 ハッカソンイベントでの活用事例

2016 年 7 月 9 日に開催されたミステリーハッカソン（http://connpass.com/event/34528/）は，UFO・超能力・UMA・奇現象・古代文明・神秘などを、ユーモアをもって情報技術によりハックするイベントであった．そこで参加者に提案フレームワークを紹介したところ，1 名の参加者（企業エンジニア，プログラミング経験者，ただし Javascript 使用経験は少ない）が活用し，開催地である津田塾大学で語り継がれる都市伝説である「津田梅子墓所参拝回数のジンクス」をテーマにしたインタラクティブ映像作品製作を行った（図 6）．これは墓所に参拝する映像の再生速度や再生位置を独自の GUI から変更したり，参拝時に参拝回数をカウント・表示しブラウザのクッキーに保存したりといった処理を行うものであった．開発および発表資料作成時間は約 7 時間であった．イベント後，提案フレームワークについてのフィードバックを自由記述で求めた．

図 6 ハッカソンでの製作事例

Figure 6 A use case at a hackathon.

良かった点として挙げられたのは，コンセプトが明確でできることがはっきりしていた点，構文がシンプルである点，時刻指定によるプログラミングが可能な点，srt.js を Dropbox の公開フォルダ等に置いて surl パラメータで読み込み指定することにより，すぐに実行・デバグできた点などであった．これは映像連動コンテンツの開発において srt.js の設計が効果的であったことを示唆するものである．

分かりにくかった点，および改善を希望する点として挙げられたのは，以下の項目である．

1. GUI 要素は HTML で記述したい．
2. 再生前に実行する「初期化セクション」がほしい．
3. ファイルの冒頭にコメントを入れたい．
4. srt の仕様として空行が意味を持つのでプログラムの可視性を高めるための空行が使えない．
5. srt の仕様として各プログラム要素に通し番号がつくが，新たな要素を挿入するときなどに困る．
6. 時間区間の重複は可能なのか．
7. 指定する時間区間は開区間なのか閉区間なのか．

1 については，現在の実装では GUI 要素もすべて Javascript から動的に追加する必要があることに起因するものである．GUI を記述した HTML を外部ファイルとして読み込む機能を今後追加したい．

2 については現状では時刻 0 秒において初期化セクションを記述することが一般的なプラクティスであるが，映像の再生まで実行されない点が問題である．時刻 0 秒に記述されているプログラムについては特別扱いし，映像の再生前に実行するなどの処理を実装することが有効かもしれない．

3 については，srt 形式の制約上現状では不可能である．一方で図 3 の 3 行目のように，Javascript 記述部の任意の行におけるダブルスラッシュによるコメントの記述が可能である．4 についても空行は使えないことが問題であるが，ダブルスラッシュのみからなる行を挿入することで空行と類似の視覚効果が得られるかもしれない．

5 については，実は多くの srt パーサにおいてこの番号の値は意味を持たず無視されるため，どんな数値でも動作に支障はない．しかしこの番号の存在はプログラマに心理的不快感を与えることは確かであるため，コードエディタ側で自動管理するなどの対策が求められる．

6 については，現状では時間区間の重複がない前提での実装になっているものの，重複の自動チェックは行っていないため協力者を混乱させたと考えられる．今後より高度なプログラミングのために重複を許す実装を検討したい．

7 について，現在すべてのプログラム要素は開始時刻と終了時刻を記述するが，実際は開始時刻にプログラム要素を実行し始めるだけであり，終了時刻において処理の終了を保証・同期させるものではない．ただし 3.1 節で述べた indexedFunction を用いる際は，あるイベントハンドラの発火受付時間として開始時刻と終了時刻が意味のある情報として活用される．indexedFunction は今回用いられなかったため，協力者が終了時間の記述に意味を見出せなかったことがこのコメントにつながったと考えられる．indexedFunction を用いない場合は終了時刻の省略もしくは自動補完が行えるような工夫が必要かもしれない．

2 以降の項目については，現状では srt.js が srt 形式に準拠することに固執することで生じてしまっている欠点である．YouTube が公式に srt 形式の字幕をサポートしているこ

とに着眼して srt.js は設計されたが，srt 形式から離れることによってよりプログラミングしやすい言語設計をすることは可能である．これは今後の課題である．

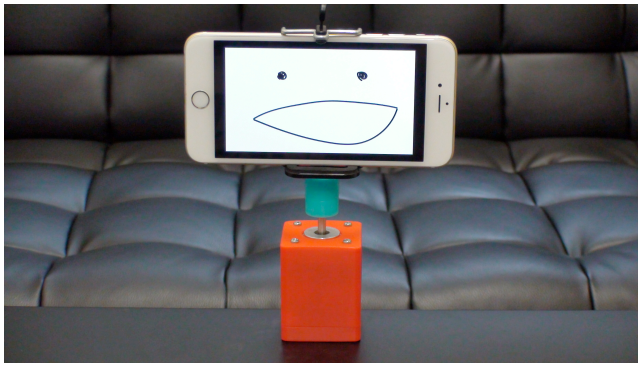

図 7　企業での製品プロトタイプ製作過程での活用事例．ロボットが向きを変えながら人間に語りかける．

Figure 7　A use case of prototyping process at a company. A robot talks to people with rotating its direction.

5.2 企業での製品プロトタイプ製作過程での活用事例

結婚支援事業を営む株式会社 Diverse では，いわゆる「婚活イベント」における ICT 技術の活用についての初期検討が行われていた．その一環として，対話型ロボットを司会役として活用するアイディアのプロトタイプ製作に提案手法が用いられた．使用者（企業研究者，プログラミング経験者，ただし Javascript 使用経験は少ない）は当初，ロボットの顔とアニメーションと音声を映像コンテンツとして作成し，それを iPhone 上で再生しながら，iPhone を搭載させた webmo の回転をノート PC 上から制御する手法をとっていた．映像コンテンツと webmo 制御の同期は手動で，すなわち映像再生ボタンを押下してすばやく制御プログラムを実行することでとっており，制御プログラムは「あるモータ制御を行ってから XX 秒後に次のモータ制御をする」というような相対時刻での記述の連鎖を用いる手法であった．

提案システムを紹介・操作説明したところ，同等のプロトタイプの製作に提案システムが採用された．開始約 15 分で一通り動作するところまで実現し，その後約 120 分が，いつどのように webmo を動かすと活き活きとした印象を与えるかなど，コンテンツのクオリティを高める作業に費やされた（図 7）．

作業後に提案フレームワークについてのフィードバックを自由記述で求めたところ，主として開発の容易さが高評価された．特に映像とプログラムの同期を手動ではなくシステムが自動で行ってくれる点が評価された．これについては再生開始時だけでなく，細部の微修正をする際，YouTube プレイヤー内のシークバーによって何度も特定の箇所を再生することで動作確認でき，コンテンツのクオリティを高める微修正作業に集中できる点で評価が高まったものである．これは相対時刻ではなく絶対時刻により実行プログラムを指定できる提案手法の大きなメリットであると考えられる．

6. 議論

6.1 脆弱性

提案システムは surl パラメータを用いて映像と連動した任意の Javascript を実行可能であるため，悪意のあるスクリプトに対し本質的に脆弱である．それに対し，以下の 2 つの対策を講じている．

- 映像の投稿者による公式の srt.js 以外が surl パラメータを用いて実行される場合には警告が表示される．
- eval()に代わるより安全度の高い実行環境である evel()が選択可能である．

今後より安全性を高めるには，ポータルサイトを公開し，そこにアカウント登録しないと srt.js を実行できないようにし，さらにそれぞれの映像コンテンツに関し不特定多数ユーザからのコメントや評価を記録することで悪質なコンテンツを排除する仕組みなどが有効であろう．

また，現在は Meshblu を用いた通信および IoT 機器との通信確立についてプレーンテキストを用いて行っているが，より厳密には RSA 公開鍵認証を使うことなどが有効であろう．

6.2 展望

6.2.1 全天周映像への対応

YouTube は近年全天周映像のアップロードと配信に対応し，スマートフォンの YouTube アプリから全天周映像を再生すると，スマートフォンの向きや角度を変えることによりあらゆる方向の映像を閲覧できるようになった．通常の映像と同様に時刻と動機したスクリプト実行はできるものの，srt.js で用いている YouTube の iframe 埋め込み API は現時点でこのスマートフォン再生機能に対応していないため，全天周映像ならではのアプリケーション開発が現時点はできない．たとえば，ホラー映像において幽霊が出現した時に恐怖感を煽る効果音を再生するなど，視野にある映像が映った場合のみ対応するプログラムを実行するような処理が実現できれば興味深いが，現状では three.js などのライブラリを用いて全天周映像を再生し，srt.js を実行するスクリプトエンジンを新たに開発し実現する必要がある．YouTube 側の今後の対応に期待したい．

6.2.2 開発支援

現状では提案システムによるコンテンツ開発は，プレーンテキストによる Javascript の編集のみによって行われる．これは開発の自由度が高い反面，プログラム初心者には参入障壁が高い．中村ら[10]，松田ら[11]および Kato ら[8]のようにブラウザ上でグラフィカルな開発を行い，内部で srt.js に変換するミドルウェアを整備すれば，よりライトなユーザにも受け入れられるであろう．

6.2.3 映像分析による srt.js 自動生成

映像を入力とし，映像認識技術や音声認識技術を用いることによってコンテンツを分析し，srt.js を自動生成するようなフィルタープログラムを開発することは興味深い研究対象である．例えば著者らが過去に開発した高速動画鑑賞システム CinemaGazer[16][17]の発話認識エンジンを用いて，映像中の発話区間のみ動画の再生速度を調整する「CinemaGazer 化スクリプト」などが自明に実現できる．さらに音声中の悲鳴やグロテスクな映像を認識できるなら，そのシーンのみを無音化したり非表示にする「ホラー対策フィルタ」が実現可能であろう．IoT と連携する例では，映像中のシーンのムードを認識することで，映像鑑賞中のユーザの部屋の照明環境について Hue などを用いて自動調整することなども可能だろう．

6.2.4 Toolification of Videos

「既存ゲームの余剰自由度で非ゲーム的目的を達成すること」として著者らが提案した Toolification of Games[15]と類似の概念を，映像コンテンツの再利用方法として応用することが考えられる．具体的には「既存映像の鑑賞の余剰自由度で非エンタテインメント的目的を達成すること」を Toolification of Videos と定義する．

たとえば渡邉らの CastOven[18]は，映像鑑賞の楽しさ，時間の流れを忘れさせる効果を活用し，電子レンジの調理時間の長さへの不満を解消する用途に用いた Toolification of Videos の事例と言える．また映像にあわせて体操や運動をするようなコンテンツはテレビの歴史とともに豊富に蓄積されているが，それらも健康増進や運動スキル獲得を図る Toolification of Videos の事例と言えるだろう．さらに，映像内に挿入される広告一般も，伝統的に用いられている Toolification of Videos である．

提案システムを用いて体操・運動コンテンツに運動を検知するセンサとの連動機能を追加することでよりインタラクティブにしたり，娯楽用途で鑑賞することを主眼に作成された映像コンテンツの価値を損なわない範囲で全く別のタスクの達成を目指す機構を備えることは有意義であろう．たとえば Tsukada らの EyeCatcher[19]のように視聴者の笑顔や泣き顔を自然に写真撮影するため，有名な喜劇・悲劇映像を用いて特定の感情を想起するシーンに連動して写真撮影することなどは提案システムを用いて容易に実現可能である．4.2 節で紹介した事例 7 はその一種である．

7. おわりに

本論文では代表的な時系列コンテンツの一つである映像コンテンツの拡張について，過去に盛んに用いられていた「時刻情報とともに実行すべきプログラムを記述するパラダイム」の有効性を再検討した．具体的には，YouTube 映像に対し公式サポートされている字幕情報のフォーマットである srt 形式のテキストファイルに Javascript を記述することによって，映像と連動したプログラムを駆動し外部 Web サービスや IoT 機器と連携することを容易に行うことができ，映像コンテンツのエンタテインメント価値を高める手法を提案した．さらに種々の具体例により本手法の有効性を示し，問題点と今後の展望について整理した．

謝辞 本研究は JSPS 科研費 JP15H02735, JP16H02867 の助成を受けたものです．

参考文献

1) 栗原一貴: 放送化の時代のフレセンテーション支援システム, 情報処理, Vol.56, No.5, pp.465-471 (2015).
2) 栗原一貴, 諸美咲: Openness-adjustable Headset：開放度を調整可能なヘッドセットの開発, WISS'15 予稿集, pp.197-198(2015).
3) Kazutaka Kurihara, David Vronay, and Takeo Igarashi: Flexible Timeline User Interface Using Constraints, In Proceedings of ACM SIGCHI'05, pp.1581-1584 (2005).
4) 渡邊恵太: 融けるデザイン, ビー・エヌ・エヌ新社, 2015.
5) 貞子 3D2 スマ４D
http://www.sadako3d.jp/suma4d/
6) +1D
http://www.plus1d.jp/
7) 後藤真孝, 吉井和佳, 藤原弘将, Matthias Mauch, 中野倫靖: Songle: ユーザが誤り訂正により貢献可能な能動的音楽鑑賞サービス, インタラクション'12, pp.1-8 (2012).
8) Jun Kato, Tomoyasu Nakano, Masataka Goto: TextAlive: Integrated Design Environment for Kinetic Typography, In Proceedings of ACM SIGCHI'15, pp.3403-3412 (2015).
9) Stepmania Gimmic
https://hackmd.io/s/Hk6p0vDY
10) 中村聡史, 石川直樹, 渡邊恵太: 個人的な小さな幸せを実現するブラウザ上での動画編集・共有手法, WISS'13 予稿集, pp.19-24 (2013).
11) 松田滉平, 中村聡史: 動画に対する音響的装飾の分析と視覚的装飾を可能とする手法の提案, エンタテインメントコンピューティングシンポジウム 2015 論文集, pp.458-464 (2015).
12) 加藤淳, 後藤 真孝: IoT アプリケーションのソフトウェア・ハードウェアを単一コードベースで開発できる統合開発環境 f3.js, インタラクション'16 予稿集, pp.132-139 (2016).
13) 馬場匠見, 橋本翔, 増井俊之: Babascript: 人とコンピュータの協調による処理実行環境, WISS'14 予稿集, pp.109-114 (2014).
14) 橋本翔, 増井俊之: GoldFish: JavaScript と Android NFC による実世界 GUI フレームワーク, インタラクション 2012 論文集, pp.867-870 (2012).
15) Kazutaka Kurihara: Toolification of Games: Achieving Non-game Purposes in the Redundant Spaces of Existing Games, In Proceedings of ACE'15, pp.31:1-31:5 (2015).
16) Kazutaka Kurihara: CinemaGazer: a System for Watching Videos at Very High Speed, In Proceedings of AVI'12, pp.108-115 (2012).
17) Kazutaka Kurihara, Yoko Sasaki, Jun Ogata, and Masataka Goto: Two-level fast-forwarding using speech detection for rapidly perusing video, In Proceedings of AH'14, pp.19:1-19:2 (2014).
18) Keita Watanabe, Shota Matsuda, Michiaki Yasumura, Masahiko Inami, and Takeo Igarashi: CastOven: a microwave oven with just-in-time video clips, In Proceedings of Ubicomp '10, pp.385-386 (2010).
19) Koji Tsukada and Maho Oki: EyeCatcher: a digital camera for capturing a variety of natural looking facial expressions in daily snapshots, In Proceedings of Pervasive' 10, Springer LNCS 6030, pp.112-129 (2010).